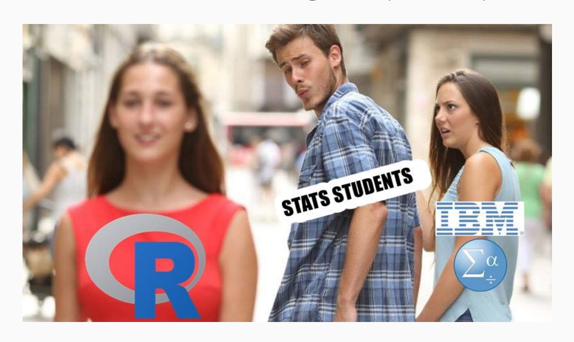
# Tema I: Introducción a R y RStudio.

Maicel Monzón

# Sumario

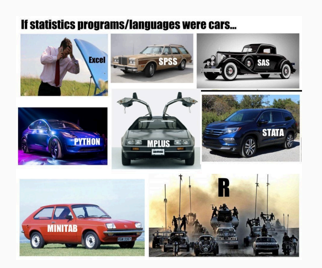
- Características generales del lenguaje
- Creación, listado y remoción de objetos en memoria
- Objetos y tipos de datos
- Obtener ayuda

R es un **lenguaje de programación** y **entorno de software libre** para **computación estadística** y **ciencia de datos** con una gran capacidad para **manipular, resumir, analizar y representar datos**.



# Uso de R

- **Google** efectividad publicitaria y pronósticos económicos
- **Facebook** analisis imágenes de perfil
- **Twitter** visualización de datos y agrupación semántica



La **Organización Mundial de la Salud (OMS)** está financiando software para R para **estimar los niveles apropiados de vitaminas y minerales** para los **programas de fortificación de alimentos**.



# Porqué debo usar R!

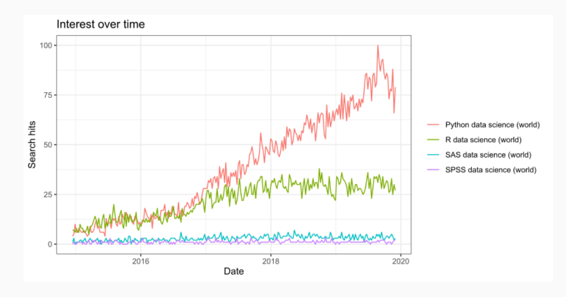
- R es un **lenguaje de programación** libre, gratuito y abierto
- Cuenta con una amplia comunidad desarrolladores
- Lista de correo (R-help)
- Blog (R-bloggers)
- Motor de búsqueda personalizado (RSeek.org)
- CRAN package repository features 15313 available packages.

- Por su gran capacidad para **leer, manipular, resumir, representar y analizar datos**: R es conocido como **"the language of data science"**



# Porqué debo usar R!

- Cuenta con la mayor gama de métodos estadísticos de alto rendimiento, robustos y validados por la comunidad científica.



# Porqué debo usar R!

- R te permite analizar cualquier clase de datos (datos rectangulares, texto, imagen, etc.) y tamaño de datos (big data).

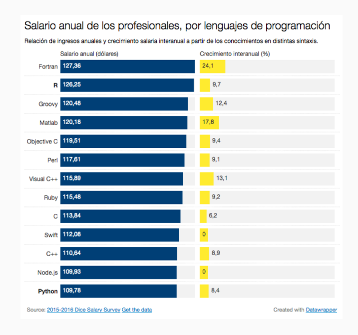



**Figure 1:** R, the language of data science

*"R está equipado para manejar datos complejos, como los de las Encuestas de Gastos y Consumo de los Hogares (HCES)."*



# R para la fortificación de alimentos a gran escala!

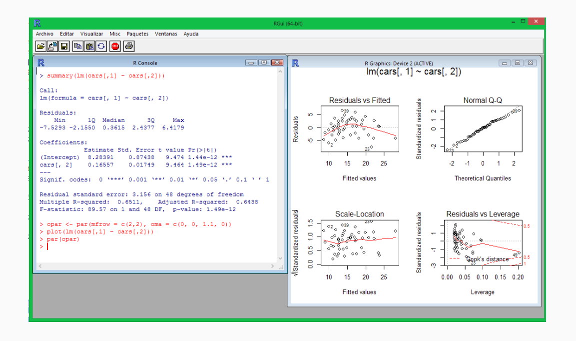
## Ejemplos



# Resumen: Elementos claves de la presentación hasta el momento

- R es un **entorno de desarrollo**
- R es un **lenguaje de programación** con **licencia GNU**
- Permite implementar procedimientos para **importar, ordenar, tranformar, analizar, representar datos de muchos tipos**

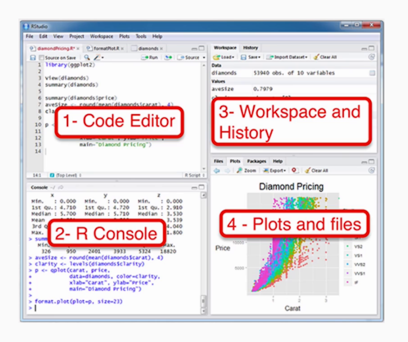
# R es un entorno de desarrollo




https://cran.r-project.org

# Rstudio

IDE (Entorno de desarrollo integrado) más popular




https://posit.co/download/rstudio-desktop/

# R es un lenguaje de programación

- **Idioma** artificial (formal)
- Diseñado para expresar **computaciones**
- Llevadas a cabo por **computadoras**

# R es un lenguaje de POO

"Un lenguaje de **programación orientado a objetos** es como un **juego de Lego**, donde creas y conectas piezas (objetos) que contienen información y acciones específicas, permitiéndote **construir programas complejos** de manera organizada y reutilizable."



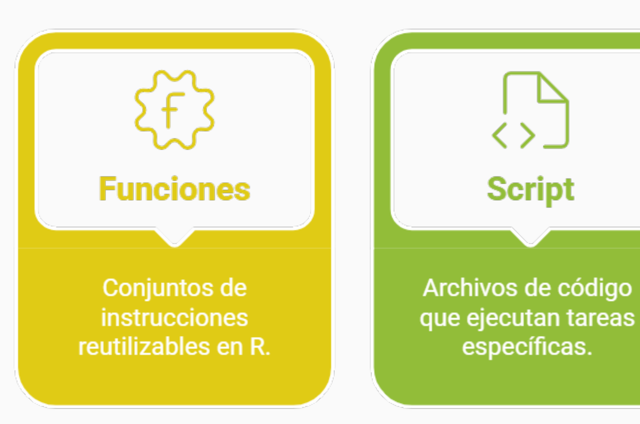
# Qué es un objeto?

"Un objeto en programación es como una pieza de Lego individual que almacena datos (como color o tamaño) y acciones propias (como unirse a otras piezas), funcionando como un bloque independiente pero conectable para crear sistemas más grandes."



# R implenta programación orientado a Objetos

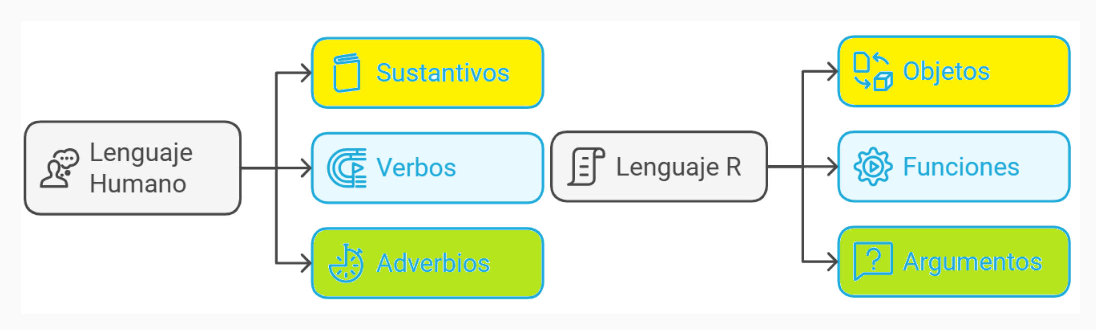
- Permite la reutilización del código (funciones,script, bibliotecas)






# R es un lenguaje similar al lenguaje humano

Los **objetos** son como **sustantivos** y las **funciones** son como **verbos**



# Objetos como sustantivos

**Definición:** Los **sustantivos** representan **cosas o conceptos**. En R, los objetos son los "sustantivos" que contienen datos.

Ejemplo: Se asignan números a una variable (Micronutrientes en Harina).

```
# Micronutrientes en un kg de Harina
hierro_por_kg <- 30     # mg de hierro
zinc_por_kg <- 15       # mg zinc
yodo_por_kg <- 50       # µg yodo
```

# Funciones como Verbos

Los **verbos** expresan **acciones** en las **lenguas humanas**. De manera similar, las **funciones** en R son los **“verbos”** que realizan **acciones** sobre los objetos.

Ejemplo asignar **(<-)** un valor a un objeto

*(Alt y -) imprime* <-

```
# # Consumo diario promedio de harina por hogar (ej: 0.5 kg)
consumo_harina <- 0.5  # kg/día
```

Si consideramos las **funciones como verbos**, los **argumentos** de las funciones pueden verse como **adverbios que modifican la acción**.

Ejemplo: La función combinar **“c()”** usa como argumentos diferentes objetos

```
# # concatenar para  crear vectores
vitaminas <- c("A", "D", "C") # vector de texto (micronutrientes)
consumo_hierro <- c(150, 200, 180)  #  vector numérico (Fe mg en 3 hogares)
```

# Combinación de Sustantivos y Verbos

**Construcción de Sentencias:** Al igual que en una oración donde los sustantivos y verbos se combinan para expresar un pensamiento completo, en R se combinan objetos (sustantivos) y funciones (verbos) para realizar operaciones.

Ejemplo: La expresión *mean(numeros)* es equivalente a **“calcular la media del consumo de hierro en la muestra”**, donde **“calcular”** es el verbo y **“el vector numérico”** es el objeto.

```
# # concatenar para  crear vectores
consumo_hierro <- c(150, 200, 180)  #  vector numérico (Fe mg en 3 hogares)
mean(x = consumo_hierro)
```

```
[1] 176.6667
```

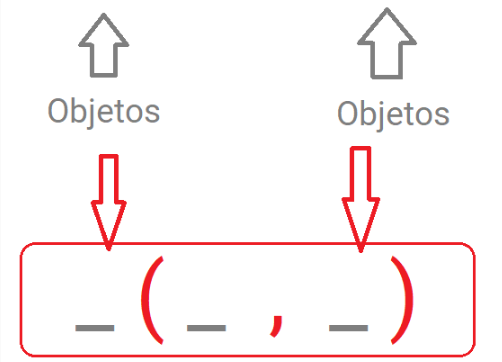
**Estructura y Sintaxis:** Al igual que el español combina sustantivos y verbos según reglas gramaticales, R combina objetos y funciones para realizar cálculos.

**Características del Lenguaje:** R tiene su propio vocabulario y sintaxis, permitiendo dar comando y estructurar soluciones a problemas

# Todo en R es un objeto y lo que sucede una función

En R, "todo lo que sucede es una llamada a función"

FUNCIÓN(*argumento 1, argumento 2,...*)




En R, "todo lo que existe en un objeto"

# Los objetos en R se crean:

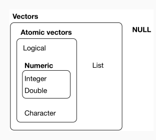
- Asignándoles un valor
- Como resultado de un cálculo
- Leyendo datos de un archivo
- etc.

# Los vectores son la estructura de datos fundamental de R base para creas otras estruturas

- vectores atómicos: elementos del mismo tipo (homogéneos)
- vectores recursivos: pueden tener elementos de varios tipos (heterogéneos)
- NULL: vector genérico de longitud cero

# Tipos de vectores atómicos

1. Lógicos: - *(TRUE o FALSE), o ( T o F)*
2. Dobles: - *forma decimal ( 0.1234), científica ( 1.23e4)*
3. Enteros: - *seguidas de L ( 1234L, 1e4L)*
4. Cadenas: - *encerrados por comillas ( “vitamina”)*



## Objeto vector atómico

la función “c()” abreviatura de **combinar** hace vectores más largos

```
anemia <- c(TRUE, FALSE)
cucharadas <- c(1L, 6L, 10L)
ingesta_hierro <- c(1, 2.5, 4.5)
micronutrientes <- c("hierro", "Ácido Ascórbico", "Zinc")
```

Cuando intentas **combinar** elementos de diferentes tipospara hacer vectores más largos ocurre la **Coerción**

```
# ingesta diaria de hierro en mg
ingesta_hierro <- c(8, 7, "11 mg")
# se convierte en vector de cadena
ingesta_hierro <- c("8", "7", "11 mg")
# las cadenas vas entre comillas
```

# Factores

- Es un vector que solo puede contener valores predefinidos
- Se utiliza para almacenar datos cualitativos o categóricos

# Factores

- Los factores se construyen a partir de vectores de enteros (atómicos) y añaden un atributo para definir los niveles (etiquetas categóricas)
- Evitan la falta de consistencia Ejemplo sexo: “Fem”, “Femenino”, “F”

Son útiles cuando conoce el conjunto de **valores posibles**, pero no todos están presentes en un conjunto de datos dados.

```
sex_char <- c(1, 2, 1)
sex_factor <- factor(x = sex_char,
                     levels = c(1, 2),
                     labels = c("masculino","femenino"))
```

```
table(sex_factor)
```

```
sex_factor
masculino  femenino
        2         1
```

## Listas

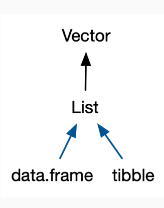
cada elemento puede ser de cualquier tipo (datos heterogenea)

construir una lista

```
# Crear lista con datos de fortificación
datos_malawi <- list(
   c("Aceite", "Azúcar", "Harina de trigo"),
   list(
    vitamina_a = c(1000, 700, 80),  # µg por 100g
    hierro = c(NA, NA, 3.0)         # mg por 100g
  ),
  c("Norte", "Centro", "Sur"),
  c(76.1, 55.6, 44.2)   # % de hogares que consumen
)
```

# Marcos de datos y tibbles

- lista con nombre de vectores (columna)
- la longitud de cada uno de sus vectores debe ser la misma a diferencia de las listas



# Marcos de datos (dataframe) creacion

Puede crear un marco de datos al proporcionar pares nombre-vector

```
library(tidyverse)
# Crear dataframe de fortificación
df_fortificacion <- tibble(
  vehiculo = c("Aceite vegetal", "Azúcar", "Harina de trigo"),
  micronutriente = c("Vitamina A", "Vitamina A", "Hierro"),
  cantidad_por_100g = c(1000, 700, 3.0), # µg para vitamina A
  cobertura = c(76.1, 55.6, 44.2), # % de hogares consumidores
  region_principal = c("Norte", "Centro", "Sur") # Distribución regional
)


str(df_fortificacion)
```

# tibbles (Subconjunto)

- Se usan operadores $ y [[

Extraer por nombre

```
# Extract by name
df_fortificacion$vehiculo
```

```
[1] "Aceite vegetal"  "Azúcar"              "Harina de trigo"
```

```
df_fortificacion[["vehiculo"]]
```

```
[1] "Aceite vegetal"  "Azúcar"              "Harina de trigo"
```

Extraer por posición

```r
# Extract by position
df_fortificacion[[2]]
```

```
[1] "Vitamina A" "Vitamina A" "Hierro"
```

- usar tubería para hacer subconjuntos

```
df_fortificacion %>% .$vehiculo
```

```
[1] "Aceite vegetal"  "Azúcar"          "Harina de trigo"
```

```
#> [1] 0.434 0.395 0.548 0.762 0.254
df_fortificacion %>% .[["vehiculo"]]
```

```
[1] "Aceite vegetal"  "Azúcar"          "Harina de trigo"
```

```
#> [1] 0.434 0.395 0.548 0.762 0.254
```

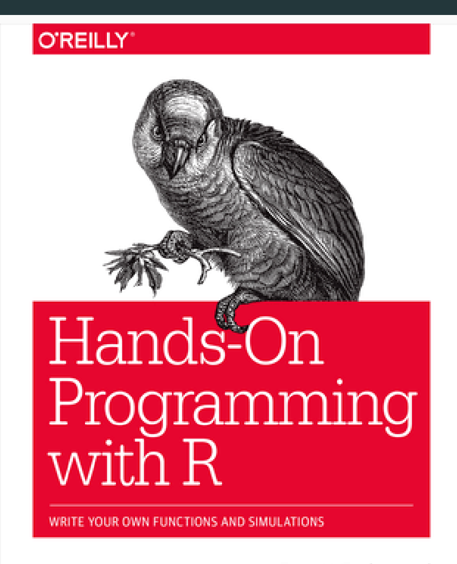
# tubería

En R, el operador %>% (tubería) permite encadenar operaciones de manera fluida, similar a cómo leerías una oración.

```
df_fortificacion %>%
  select(region_principal) %>%
  count()
```

```
# A tibble: 1 x 1
      n
  <int>
1     3
```

# Bibliografía




disponible:

# Próxima actividad: Desafio 1. Hierro “Cimientos de Código”

